

How would you refer to the array element 1 in the above array? The first subscript should be [2], since the element is in third two-dimensional array; the second subscript should be [3] since the element is in fourth row of the two-dimensional array; and the third subscript should be [1] since the element is in second position in the one-dimensional array. We can therefore say that the element 1 can be referred as **arr[2][3][1]**. It may be noted here that the counting of array elements even for a 3-D array begins with zero. Can we not refer to this element using pointer notation? Of course, yes. For example, the following two expressions refer to the same element in the 3-D array:

```
arr[2][3][1]
*( *( *( arr + 2 ) + 3 ) + 1 )
```

Summary

- (a) An array is similar to an ordinary variable except that it can store multiple elements of similar type.
- (b) Compiler doesn't perform bounds checking on an array.
- (c) The array variable acts as a pointer to the zeroth element of the array. In a 1-D array, zeroth element is a single value, whereas, in a 2-D array this element is a 1-D array.
- (d) On incrementing a pointer it points to the next location of its type.
- (e) Array elements are stored in contiguous memory locations and so they can be accessed using pointers.
- (f) Only limited arithmetic can be done on pointers.

Exercise

Simple arrays

[A] What would be the output of the following programs:

- (a) `main()`

```

{
    int num[26], temp ;
    num[0] = 100 ;
    num[25] = 200 ;
    temp = num[25] ;
    num[25] = num[0] ;
    num[0] = temp ;
    printf ( "\n%d %d", num[0], num[25] ) ;
}

```

OUTPUT
200 100

(b) main()

```

{
    int array[26], i ;
    for ( i = 0 ; i <= 25 ; i++ )
    {
        array[i] = 'A' + i ;
        printf ( "\n%d %c", array[i], array[i] ) ;
    }
}

```

OUTPUT
65 A
66 B
.
.
89 Y
90 Z
<-- upto 90 Z

(c) main()

```

{
    int sub[50], i ;
    for ( i = 0 ; i <= 48 ; i++ )
    {
        sub[i] = i ;
        printf ( "\n%d", sub[i] ) ;
    }
}

```

OUTPUT
49

[B] Point out the errors, if any, in the following program segments:

(a) /* mixed has some char and some int values */
int char mixed[100] ;
 main()
 {
 int a[10], i ;

```
    for ( i = 1 ; i <= 10 ; i++ )
    {
        scanf ( "%d", a[i] ) ;
        printf ( "%d", a[i] ) ;
    }
}
```

```
(b) main()
{
    int size ;
    scanf ( "%d", &size ) ;
    int arr[size] ; ✗
    for ( i = 1 ; i <= size ; i++ )
    {
        scanf ( "%d", arr[i] ) ;
        printf ( "%d", arr[i] ) ;
    }
}
```

```
(c) main()
{
    int i, a = 2, b = 3 ;
    int arr[ 2 + 3 ] ;
    for ( i = 0 ; i < a+b ; i++ )      NO error
    {
        scanf ( "%d", &arr[i] ) ;
        printf ( "\n%d", arr[i] ) ;
    }
}
```

[C] Answer the following:

(a) An array is a collection of

1. different data types scattered throughout memory
2. the same data type scattered throughout memory
3. the same data type placed next to each other in memory
4. different data types placed next to each other in memory

(b) Are the following array declarations correct?

int a (25) ; ✗
int size = 10, b[size] ; ✗
int c = {0,1,2} ; ✗

(c) Which element of the array does this expression reference?

num[4] References the fifth element of the array num

(d) What is the difference between the 5's in these two expressions? (Select the correct answer)

- a) int num[5] ; a) declares an array num of size 5, meaning it can hold 5 elements
b) num[5] = 11 ; b) it set the value of of sixth element to 11

1. first is particular element, second is type
2. first is array size, second is particular element
3. first is particular element, second is array size
4. both specify array size

(e) State whether the following statements are True or False:

1. The array **int num[26]** has twenty-six elements. ✓
2. The expression **num[1]** designates the first element in the array ✗
3. It is necessary to initialize the array at the time of declaration. ✗
4. The expression **num[27]** designates the twenty-eighth element in the array. ✗

[D] Attempt the following:

- (a) Twenty-five numbers are entered from the keyboard into an array. The number to be searched is entered through the keyboard by the user. Write a program to find if the number to be searched is present in the array and if it is present, display the number of times it appears in the array.

- (b) Twenty-five numbers are entered from the keyboard into an array. Write a program to find out how many of them are positive, how many are negative, how many are even and how many odd.
- (c) Implement the Selection Sort, Bubble Sort and Insertion sort algorithms on a set of 25 numbers. (Refer Figure 8.11 for the logic of the algorithms)
- Selection sort
 - Bubble Sort
 - Insertion Sort

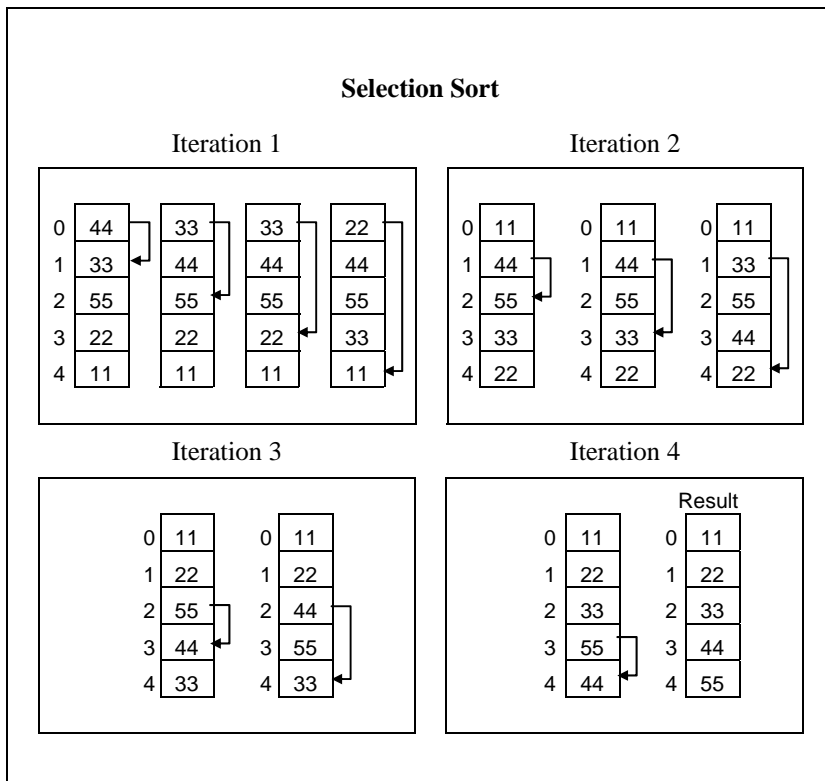


Figure 8.11 (a)

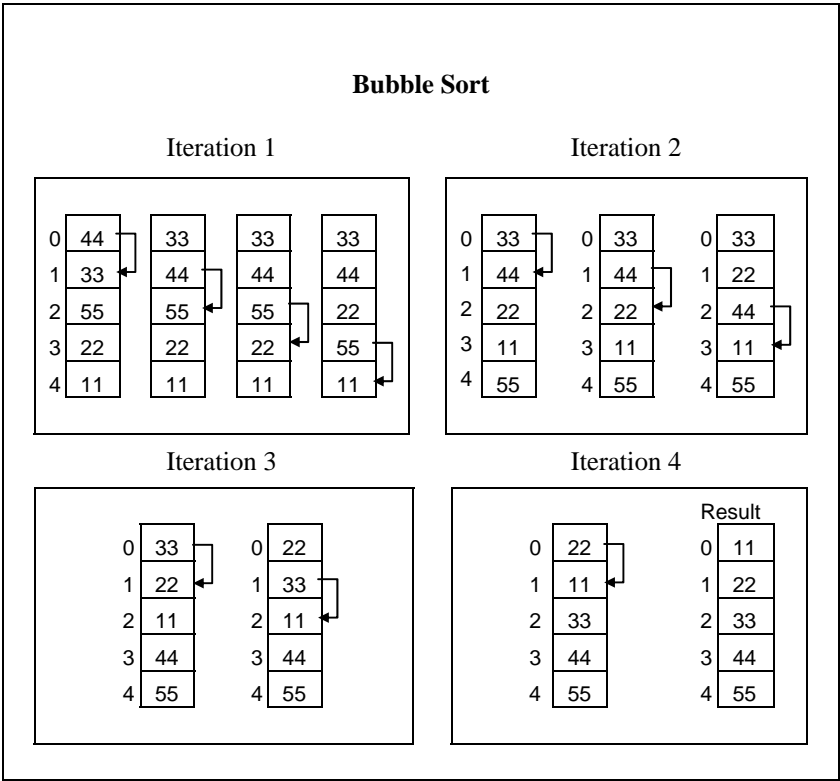


Figure 8.11 (b)

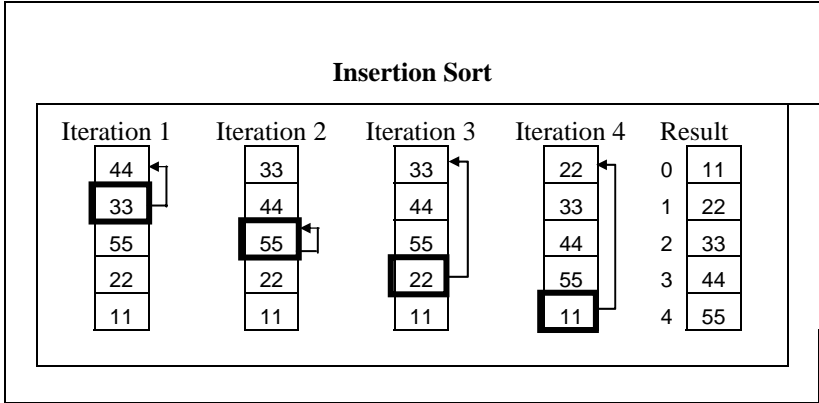


Figure 8.11 (c)

- (d) Implement the following procedure to generate prime numbers from 1 to 100 into a program. This procedure is called sieve of Eratosthenes.

- step 1 Fill an array **num[100]** with numbers from 1 to 100
- step 2 Starting with the second entry in the array, set all its multiples to zero.
- step 3 Proceed to the next non-zero element and set all its multiples to zero.
- step 4 Repeat step 3 till you have set up the multiples of all the non-zero elements to zero
- step 5 At the conclusion of step 4, all the non-zero entries left in the array would be prime numbers, so print out these numbers.

More on arrays, Arrays and pointers

[E] What would be the output of the following programs:

(a)

```
main()
{
    int b[] = { 10, 20, 30, 40, 50 };
    int i;
    for ( i = 0 ; i <= 4 ; i++ )
        printf ( "\n%d" *( b + i ) );
}
```

OUTPUT

10
20
30
40
50

(b)

```
main()
{
    int b[] = { 0, 20, 0, 40, 5 };
    int i, *k;
    k = b;
    for ( i = 0 ; i <= 4 ; i++ )
    {
        printf ( "\n%d" *k );
    }
}
```

OUTPUT

0
20
0
40
5

```

        k++ ;
    }
}

(c)  main()
    {
        int a[] = { 2, 4, 6, 8, 10 } ;
        int i ;
        change ( a, 5 ) ;
        for ( i = 0 ; i <= 4 ; i++ )
            printf( "\n%d", a[i] ) ;
    }
?→ change ( int *b, int n )
    {
        int i ;
        for ( i = 0 ; i < n ; i++ )
            *( b + i ) = *( b + i ) + 5 ;
    }

(d)  main()
    {
        int a[5], i, b = 16 ;
        for ( i = 0 ; i < 5 ; i++ )
            a[i] = 2 * i ;
        f ( a, b ) ;
        for ( i = 0 ; i < 5 ; i++ )
            printf ( "\n%d", a[i] ) ;
            printf( "\n%d", b ) ;
    }
    f ( int *x, int y )
    {
        int i ;
        for ( i = 0 ; i < 5 ; i++ )
            *( x + i ) += 2 ;
        y += 2 ;
    }

```

if its void change()
then

7
9
11
13
15

OUTPUT

2
4
6
8
10
16


```
(e) main()
{
    static int a[5];
    int i;
    for ( i = 0 ; i <= 4 ; i++ )
        printf ( "\n%d", a[i] );
}
```

OUTPUT

0
0
0
0
0

```
(f) main()
{
    int a[5] = { 5, 1, 15, 20, 25 };
    int i, j, k = 1, m;
    i = ++a[1];
    j = a[1]++;
    m = a[i++];
    printf ( "\n%d %d %d", i, j, m );
}
```

OUTPUT

3 2 15

[F] Point out the errors, if any, in the following programs:

```
(a) main()
{
    int array[6] = { 1, 2, 3, 4, 5, 6 };
    int i;
    for ( i = 0 ; i <= 25 ; i++ )
        printf ( "\n%d", array[i] );
}
```

array size is 6, can't print more than 6
elements, ERROR

```
(b) main()
{
    int sub[50], i;
    for ( i = 1 ; i <= 50 ; i++ )
    {
        sub[i] = i;
        printf ( "\n%d", sub[i] );
    }
}
```

- (c)

```
main()
{
    int a[] = { 10, 20, 30, 40, 50 };
    int j;
    j = a; /* store the address of zeroth element */
    j = j + 3;
    printf ( "\n%d" *j );
}
```

Assigned as int but stored pointer
- (d)

```
main()
{
    float a[] = { 13.24, 1.5, 1.5, 5.4, 3.5 };
    float *j;
    j = a;
    j = j + 4;
    printf ( "\n%d %d %d", j, *j, a[4] );
}
```
- (e)

```
main()
{
    float a[] = { 13.24, 1.5, 1.5, 5.4, 3.5 };
    float *j, *k;
    j = a;
    k = a + 4;
    j = j * 2;
    k = k / 2;
    printf ( "\n%d %d", *j, *k );
}
```
- (f)

```
main()
{
    int max = 5;
    float arr[max];
    for ( i = 0 ; i < max ; i++ )
        scanf ( "%f", &arr[i] );
}
```

? Not Declared

[G] Answer the following:

- (a) What would happen if you try to put so many values into an array when you initialize it that the size of the array is exceeded?
1. nothing
 2. possible system malfunction
 3. error message from the compiler
 4. other data may be overwritten
- (b) In an array `int arr[12]` the word `arr` represents the a name of the array
- (c) What would happen if you put too few elements in an array when you initialize it?
1. nothing
 2. possible system malfunction
 3. error message from the compiler
 4. unused elements will be filled with 0's or garbage
- (d) What would happen if you assign a value to an element of an array whose subscript exceeds the size of the array?
1. the element will be set to 0
 2. nothing, it's done all the time
 3. other data may be overwritten
 4. error message from the compiler
- (e) When you pass an array as an argument to a function, what actually gets passed?
1. address of the array
 2. values of the elements of the array
 3. address of the first element of the array
 4. number of elements of the array


(f) Which of these are reasons for using pointers?

1. To manipulate parts of an array
2. To refer to keywords such as **for** and **if**
3. To return more than one value from a function
4. To refer to particular programs more conveniently


(g) If you don't initialize a static array, what would be the elements set to?

1. 0
2. an undetermined value
3. a floating point number
4. the character constant '\0'

[H] State True or False:


(a) Address of a floating-point variable is always a whole number. 

(b) Which of the following is the correct way of declaring a float pointer:

5. float ptr ;
6. float *ptr ; 
7. *float ptr ;
8. None of the above

(c) Add the missing statement for the following program to print 35.

```
main()  
{  
    int j, *ptr ;  
    *ptr = 35 ;  
    printf ( "\n%d", j ) ;  
}
```

 `*ptr = &j;`
`*ptr = 35;`

- (d) if **int s[5]** is a one-dimensional array of integers, which of the following refers to the third element in the array?

- 9. `*(s + 2)`
- 10. `*(s + 3)`
- 11. `s + 3`
- 12. `s + 2`

[I] Attempt the following:

- (a) Write a program to copy the contents of one array into another in the reverse order.
- (b) If an array **arr** contains **n** elements, then write a program to check if **arr[0] = arr[n-1]**, **arr[1] = arr[n-2]** and so on.
- (c) Find the smallest number in an array using pointers.
- (d) Write a program which performs the following tasks:
 - initialize an integer array of 10 elements in **main()**
 - pass the entire array to a function **modify()**
 - in **modify()** multiply each element of array by 3
 - return the control to **main()** and print the new array elements in **main()**
- (e) The screen is divided into 25 rows and 80 columns. The characters that are displayed on the screen are stored in a special memory called VDU memory (not to be confused with ordinary memory). Each character displayed on the screen occupies two bytes in VDU memory. The first of these bytes contains the ASCII value of the character being displayed, whereas, the second byte contains the colour in which the character is displayed.

For example, the ASCII value of the character present on zeroth row and zeroth column on the screen is stored at

location number 0xB8000000. Therefore the colour of this character would be present at location number 0xB8000000 + 1. Similarly ASCII value of character in row 0, col 1 will be at location 0xB8000000 + 2, and its colour at 0xB8000000 + 3.

With this knowledge write a program which when executed would keep converting every capital letter on the screen to small case letter and every small case letter to capital letter. The procedure should stop the moment the user hits a key from the keyboard.

This is an activity of a rampant Virus called Dancing Dolls. (For monochrome adapter, use 0xB0000000 instead of 0xB8000000).

More than one dimension

[J] What would be the output of the following programs:

```
(a)  main()
    {
        int n[3][3] = {
                                2, 4, 3,
                                6, 8, 5,
                                3, 5, 1
                            };
        printf ( "\n%d %d %d", *n, n[3][3], n[2][2] );
    }
```

```
(b)  main()
    {
        int n[3][3] = {
                                2, 4, 3,
                                6, 8, 5,
                                3, 5, 1
                            };
        int i, *ptr ;
```

```
    ptr = n ;
    for ( i = 0 ; i <= 8 ; i++ )
        printf ( "\n%d", *( ptr + i ) ) ;
}

(c)  main()
    {
        int n[3][3] = {
                                2, 4, 3,
                                6, 8, 5,
                                3, 5, 1
                            };

        int i, j ;
        for ( i = 0 ; i <= 2 ; i++ )
            for ( j = 0 ; j <= 2 ; j++ )
                printf ( "\n%d %d", n[i][j], *( *( n + i ) + j ) ) ;
    }
```

[K] Point out the errors, if any, in the following programs:

```
(a)  main()
    {
        int twod[ ][ ] = {
                                2, 4,
                                6, 8
                            };

        printf ( "\n%d", twod ) ;
    }

(b)  main()
    {
        int three[3][ ] = {
                                2, 4, 3,
                                6, 8, 2,
                                2, 3, 1
                            };

        printf ( "\n%d", three[1][1] ) ;
    }
```

```
}
```

[L] Attempt the following:

- (a) How will you initialize a three-dimensional array **threed[3][2][3]**? How will you refer the first and last element in this array?
- (b) Write a program to pick up the largest number from any 5 row by 5 column matrix.
- (c) Write a program to obtain transpose of a 4 x 4 matrix. The transpose of a matrix is obtained by exchanging the elements of each row with the elements of the corresponding column.
- (d) Very often in fairs we come across a puzzle that contains 15 numbered square pieces mounted on a frame. These pieces can be moved horizontally or vertically. A possible arrangement of these pieces is shown below:

1	4	15	7
8	10	2	11
14	3	6	13
12	9	5	

Figure 8.12

As you can see there is a blank at bottom right corner. Implement the following procedure through a program:

Draw the boxes as shown above. Display the numbers in the above order. Allow the user to hit any of the arrow keys (up, down, left, or right).

If user hits say, right arrow key then the piece with a number 5 should move to the right and blank should replace the original position of 5. Similarly, if down arrow key is hit, then 13 should move down and blank should replace the original position of 13. If left arrow key or up arrow key is hit then no action should be taken.

The user would continue hitting the arrow keys till the numbers aren't arranged in ascending order.

Keep track of the number of moves in which the user manages to arrange the numbers in ascending order. The user who manages it in minimum number of moves is the one who wins.

How do we tackle the arrow keys? We cannot receive them using **scanf()** function. Arrow keys are special keys which are identified by their 'scan codes'. Use the following function in your program. It would return the scan code of the arrow key being hit. Don't worry about how this function is written. We are going to deal with it later. The scan codes for the arrow keys are:

up arrow key – 72 down arrow key – 80
left arrow key – 75 right arrow key – 77

```
/* Returns scan code of the key that has been hit */
#include "dos.h"
getkey()
{
    union REGS i, o ;
```

```

while ( !kbhit() )
    ;
i.h.ah = 0 ;
int86 ( 22, &i, &o ) ;
return ( o.h.ah ) ;
}

```

(e) Those readers who are from an Engineering/Science background may try writing programs for following problems.

- (1) Write a program to add two 6 x 6 matrices.
- (2) Write a program to multiply any two 3 x 3 matrices.
- (3) Write a program to sort all the elements of a 4 x 4 matrix.
- (4) Write a program to obtain the determinant value of a 5 x 5 matrix.

(f) Match the following with reference to the following program segment:

```

int i, j, = 25;
int *pi, *pj = & j;
.....
..... /* more lines of program */
.....
*pj = j + 5;
j = *pj + 5 ;
pj = pj ;
*pi = i + j

```

Each integer quantity occupies 2 bytes of memory. The value assigned to **i** begin at (hexadecimal) address F9C and the value assigned to **j** begins at address F9E. Match the value represented by left hand side quantities with the right.

- | | |
|------------|------------|
| 1. &i | a. 30 |
| 2. &j | b. F9E |
| 3. pj | c. 35 |
| 4. *pj | d. FA2 |

- | | | | |
|-----|--------------|----|-------------|
| 5. | i | e. | F9C |
| 6. | pi | f. | 67 |
| 7. | *pi | g. | unspecified |
| 8. | (pi + 2) | h. | 65 |
| 9. | (*pi + 2) | i. | F9E |
| 10. | * (pi + 2) | j. | F9E |
| | | k. | FAO |
| | | l. | F9D |

(g) Match the following with reference to the following segment:

```
int x[3][5] = {
    { 1, 2, 3, 4, 5 },
    { 6, 7, 8, 9, 10 },
    { 11, 12, 13, 14, 15 }
}, *n = &x ;
```

- | | | | |
|-----|----------------------|----|----|
| 1. | *(*(x + 2) + 1) | a. | 9 |
| 2. | *(*x + 2) + 5 | b. | 13 |
| 3. | *(*(x + 1)) | c. | 4 |
| 4. | *(*(x) + 2) + 1 | d. | 3 |
| 5. | * (*(x + 1) + 3) | e. | 2 |
| 6. | *n | f. | 12 |
| 7. | *(n + 2) | g. | 14 |
| 8. | *(n + 3) + 1 | h. | 7 |
| 9. | *(n + 5)+1 | i. | 1 |
| 10. | ++*n | j. | 8 |
| | | k. | 5 |
| | | l. | 10 |
| | | m. | 6 |

(h) Match the following with reference to the following program segment:

```
struct
{
    int x, y;
} s[] = { 10, 20, 15, 25, 8, 75, 6, 2 };
int *i ;
i = s ;
```

1.	$*(i + 3)$	a.	85
2.	$s[i[7]].x$	b.	2
3.	$s[(s + 2) \rightarrow y / 3[I]].y$	c.	6
4.	$i[i[1] - i[2]]$	d.	7
5.	$i[s[3].y]$	e.	16
6.	$(s + 1) \rightarrow x + 5$	f.	15
7.	$*(1 + i) ** (i + 4) / *i$	g.	25
8.	$s[i[0] - i[4]].y + 10$	h.	8
9.	$(*(s + *(i + 1) / *i)).x + 2$	i.	1
10.	$++i[i[6]]$	j.	100
		k.	10
		l.	20

- (i) Match the following with reference to the following program segment:

```
unsigned int arr[3][3] = {
    2, 4, 6,
    9, 1, 10,
    16, 64, 5
};
```

1.	$**arr$	a.	64
2.	$**arr < *(*arr + 2)$	b.	18
3.	$*(arr + 2) / (*(*arr + 1) > **arr)$	c.	6
4.	$*(arr[1] + 1) arr[1][2]$	d.	3
5.	$*(arr[0]) *(arr[2])$	e.	0
6.	$arr[1][1] < arr[0][1]$	f.	16
7.	$arr[2][[1] \& arr[2][0]$	g.	1
8.	$arr[2][2] arr[0][1]$	h.	11
9.	$arr[0][1] ^ arr[0][2]$	i.	20
10.	$++**arr + --arr[1][1]$	j.	2
		k.	5
		l.	4

- (j) Write a program that interchanges the odd and even components of an array.
- (k) Write a program to find if a square matrix is symmetric.

- (l) Write a function to find the norm of a matrix. The norm is defined as the square root of the sum of squares of all elements in the matrix.
- (m) Given an array **p[5]**, write a function to shift it circularly left by two positions. Thus, if $p[0] = 15$, $p[1] = 30$, $p[2] = 28$, $p[3] = 19$ and $p[4] = 61$ then after the shift $p[0] = 28$, $p[1] = 19$, $p[2] = 61$, $p[3] = 15$ and $p[4] = 30$. Call this function for a (4 x 5) matrix and get its rows left shifted.
- (n) A 6 x 6 matrix is entered through the keyboard and stored in a 2-dimensional array **mat[7][7]**. Write a program to obtain the Determinant values of this matrix.
- (o) For the following set of sample data, compute the standard deviation and the mean.

-6, -12, 8, 13, 11, 6, 7, 2, -6, -9, -10, 11, 10, 9, 2

The formula for standard deviation is

$$\frac{\sqrt{(x_i - \bar{x})^2}}{n}$$

where x_i is the data item and \bar{x} is the mean.

- (p) The area of a triangle can be computed by the sine law when 2 sides of the triangle and the angle between them are known.

Area = (1 / 2) ab sin (angle)

Given the following 6 triangular pieces of land, write a program to find their area and determine which is largest,

Plot No.	a	b	angle
1	137.4	80.9	0.78
2	155.2	92.62	0.89
3	149.3	97.93	1.35

4	160.0	100.25	9.00
5	155.6	68.95	1.25
6	149.7	120.0	1.75

- (q) For the following set of n data points (x, y) , compute the correlation coefficient r , given by

$$r = \frac{\sum xy - \sum x \sum y}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}$$

x	y
34.22	102.43
39.87	100.93
41.85	97.43
43.23	97.81
40.06	98.32
53.29	98.32
53.29	100.07
54.14	97.08
49.12	91.59
40.71	94.85
55.15	94.65

- (r) For the following set of point given by (\mathbf{x}, \mathbf{y}) fit a straight line given by

$$y = a + bx$$

where,

$$a = \bar{y} - b\bar{x} \quad \text{and}$$

$$b = \frac{n \sum yx - \sum x \sum y}{[n \sum x^2 - (\sum x)^2]}$$

x	y
3.0	1.5

4.5	2.0
5.5	3.5
6.5	5.0
7.5	6.0
8.5	7.5
8.0	9.0
9.0	10.5
9.5	12.0
10.0	14.0

- (s) The **X** and **Y** coordinates of 10 different points are entered through the keyboard. Write a program to find the distance of last point from the first point (sum of distance between consecutive points).